



# Faculty of Engineering

# MEng in Engineering Design

## YEAR 4 PROJECT

# THE AUTOMATED DISASSEMBLY OF LITHIUM-ION BATTERY MODULES WITH COMPUTER VISION

**Felix Newport-Mangell**

Project thesis submitted in support of the degree of

Master of Engineering

Project Advisor: Mervyn White, Department of Civil Engineering

May 2022

# Acknowledgements

For his shrewd advice in shaping the objectives of this research, his dependable support, and genuine interest in the work, I would like to thank my project supervisor, Mervyn White.

I would like to thank Jack Pearson of Remix Robotics for helping establish the objectives of the group project, and his feedback throughout this research.

Thanks to Amaury Carmignac for his provision of battery modules used in the central research project of this report. Thanks also go to James Irvin and Adam Honnywill for their participation the time trials, report writing advice and constructive feedback.

I would also like to thank Paul Harper, for his unwavering dedication to all things Engineering Design, and for his outstanding effort to support every one of his students.

Finally, thanks also to the creators of Python, to Ultralytics and the open-source community for building the programming libraries central to the workings of the technology developed in this research.

DECLARATION

The accompanying research project report entitled: THE AUTOMATED DISASSEMBLY OF LITHIUM-ION BATTERY MODULES WITH COMPUTER VISION is submitted in the fourth year of study towards an application for the degree of Master of Engineering in Engineering Design at the University of Bristol. The report is based upon independent work by the candidate. All contributions from others have been acknowledged above. The views expressed within the report are those of the author and not of the University of Bristol.

I hereby declare that the above statements are true.

Signed (author)

……..

Full Name

Felix Newport-Mangell

Date

27th April 2022

# Executive Summary

Increased worldwide demand for clean technology is rapidly driving production and uptake of electric vehicles. By 2030 it is predicted that electric vehicles will account for 32% of 30 million new car sales. This increase in vehicles will be matched with an increase in lithium-ion battery (LIB) waste as these vehicles reach their end of life - fewer than 5% of LIBs are currently recycled.

Re-use is preferred to recycling, as cells are often recycled for raw materials despite meeting specifications for other applications, such as stationary storage, which requires batteries to be disassembled to cells. At present, disassembly is done manually but is too expensive to be economically viable at scale. There is therefore an opportunity to develop cost-reducing processes and technologies that incentivise disassembly and re-use through a cascade of applications. In collaboration with Remix Robotics, this work is part of a group research project that focused on the use of robotics and automation for end-of-life disassembly of LIBs.

This individual research project aimed to develop an automated system to assist in the disassembly of lithium-ion battery modules to cells. Early research identified key challenges in developing an automated system that could disassemble all modules on the market. There is no standardisation of design for battery packs, modules, or cells within the automotive sector, resulting in a considerable diversity of products that cannot be disassembled with the traditional technology of 'stiff' disassembly stations, which are optimised for a specific set of pre-programmed tasks. This provides an opportunity for the implementation of robotic systems that are able to perceive their environment and make intelligent decisions on how best to disassemble LIBs.

Computer Vision (CV) has broad applications in robotic perception. Applied to the broader research aim, it was decided to investigate the feasibility of using two CV methods to detect welded cell connections on cylindrical cell modules: a Hough Circle Transform implementation through OpenCV; and a neural network object detection model, YOLOv5s. For this purpose, images of four unique modules were captured and labelled to train the neural network.

Following a series of experiments, optimisation, and evaluation, the YOLOv5s model was taken forward and tested in its ability to generalise to module design that it had not been trained on. Hyperparameter optimisation techniques significantly improved the model's prediction accuracy on modules included in the training data but degraded in performance when generalising to the unseen module. By identifying the cause of this result as a case of model overfitting, it was possible to produce a model that was capable of locating all connections within a 3mm precision threshold on a module that it had not encountered during training, at more than 10 times faster than a human labeller. Without any manufacturer's data, manual instruction, or rigid disassembly systems, a vision system based on the methods developed in this proof-of-concept would be fully autonomous in the disassembly of cell connections.

The results of this research are an encouraging step in the design of a fully automated system for the disassembly of cylindrical cell modules. The output of the proof-of-concept developed is a set of co-ordinates relative to the module image. Further work to integrate the co-ordinates into a traversable path, a system to control the Z-axis location of the effector, and an investigation into effector design for connection removal, are the critical next steps of full system development, and could be objectives of the 5$^{th}$ year group project.

# Table of Contents

# List of Figures



# List of Tables

# 1. Introduction

The ongoing transition to Electric Vehicles (EVs) is a promising sign of progress towards the widespread adoption of sustainable technologies. EVs powered by electricity generated from renewable sources will greatly reduce the environmental impact of transport. Despite this promise, the materials used in the most prevalent cell type – lithium-ion (li-ion) – are finite, and the process of extracting the raw materials used in lithium-ion battery LIB is both energy intensive and environmentally damaging [1].

By 2030, it is estimated that there will be 30 million EVs on European roads and over 100 million worldwide [2]. This surge in demand will create similar surges in demand for EV batteries, currently dominated by LIBs. It is estimated that more than one million electric cars were sold in 2017, resulting in 250,000 tons of battery packs that will be potentially discarded in the near future [3].

In the waste management hierarchy, re-use is considered preferable to recycling. Because considerable value is embedded in manufactured LIBs, it has been suggested that their use should be cascaded through a hierarchy of applications to optimize material use and life-cycle impacts [4]. This requires battery packs to be disassembled at each stage in the application hierarchy, which is presently done manually by trained specialists.

## 1.1. Group aims

The focus of the 5th year group project will be to develop a process or technology to enable the commercial reuse of EV batteries [5]. This can be made possible by developing robotic systems that could eliminate the risk of harm to workers and reduce the cost of labour, raising the economic incentive to disassemble batteries.

Following from a period of early research, project opportunities were discussed as a group and an outline of the problem was constructed. The result was to divide individual research focus into 4 streams:

1. Market research and business strategy
2. Assessment and process monitoring
3. Disassembly from battery pack to module level
4. Disassembly from module to cell level

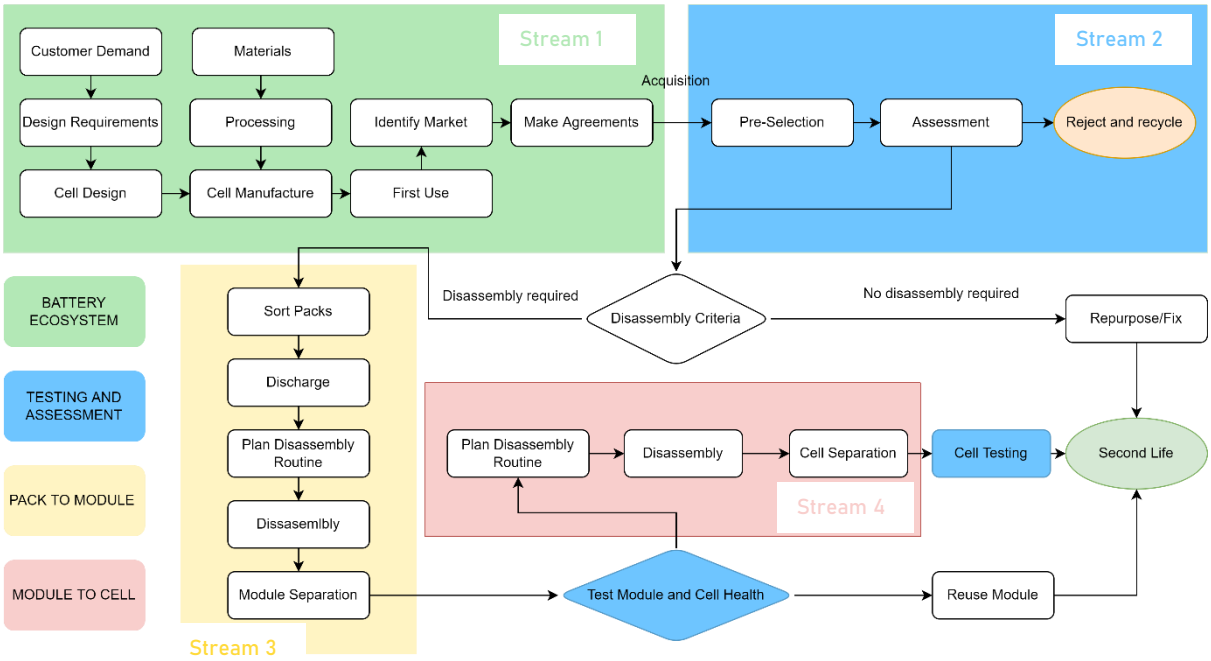


*Figure 1 Breakdown of individual research project focus and relation to group aims*

## 1.2. Individual Project Aim

This project investigates the process of developing an automated system to assist in the disassembly of lithium-ion battery modules to cells, stream 4.

A fully automated system would be able to take any battery module on the market, then safely, quickly, and cost-effectively output all cells in optimal condition whilst minimising waste from the disassembly process. This system would require sophisticated and intelligent robotics that are an ongoing domain of academic and industrial research. It was realised that it would be necessary to constrain the scope of the problem addressed by this project.

The research is separated into two stages. Prior to phase 1, the initial discovery phase, uncertainty and unknowns encountered made it challenging to identify any one area of battery module disassembly worth further investigation. The objective of phase 1 was to identify a critical area of battery module disassembly that merits investigation, to help enable the overall project aim. Then in phase 2, it would be possible to evaluate the feasibility of automating processes related to the critical area, and hypotheses of how it might be implemented in practice could be developed.

Phase 1

1. Develop an understanding of the following aspects of LIB and module design and disassembly:
    a. Design of modules, their commonalities, and differences
    b. Disassembly procedures and tooling requirements
    c. Existing research in automated module disassembly
2. Theory and techniques of automated disassembly:
    a. In structured environments
    b. In uncertain environments
    c. With product complexity and variety
3. Identify the objective for phase 2, building on existing research

Upon completion of phase 1, it was decided that the objective of phase 2 would be to investigate the feasibility of using computer vision as a component of a motion planner system for the disassembly of welded connections on cylindrical cell modules. The justification for this decision is presented in section 3.

Phase 2

1. Research into computer vision methods and related tools
2. Propose methods to locate welded connections
3. Develop an evaluation metric to classify the performance of different methods
4. Acquire images of modules
5. Implement methods to locate welded connections
    a. Evaluate performance
    b. Optimise performance
    c. Final method down-selection, implementation, and evaluation
6. Develop motion planner based on inferred data
7. Integrate motion planner within a simulator

These research phases will be covered sequentially in this report, with phase 1 forming the literature review, and phase 2 the methodology and experimentation (other than phase 2 objective 1 which is found in the literature review). The end result is a proof-of-concept computer vision system capable of identifying the location of welded connections on cylindrical cell modules. This is achieved using a deep learning object detection model, YOLOv5s, trained on images of 4 unique module designs. The ability of the model to generalise to unseen module designs is verified by training the model on images of 3 module designs, then tested on the fourth. The results of this research are summarised and discussed, and recommendations for further research and development are made.

# 2. Literature Review

A broad range of literature was explored, analysed, and its key findings condensed into three main fields: LIB modules and their disassembly, automated disassembly, and computer vision (CV). The first two of these topics were researched during phase 1, to inform objective selection for phase 2. Research in computer vision took place after the central research objective of this project was defined.

## 2.1. Lithium-Ion Battery Modules and Their Disassembly

### 2.1.1. Module Design

In EVs, modules are connected to one another to form part of the battery, which are themselves constructed from cells. EV battery modules are made from cylindrical, prismatic, or pouch cells. In a battery, the same cell types constitute the modules. Figure 2 provides a graphic deconstructing the battery pack to cells for 3 popular EV models.

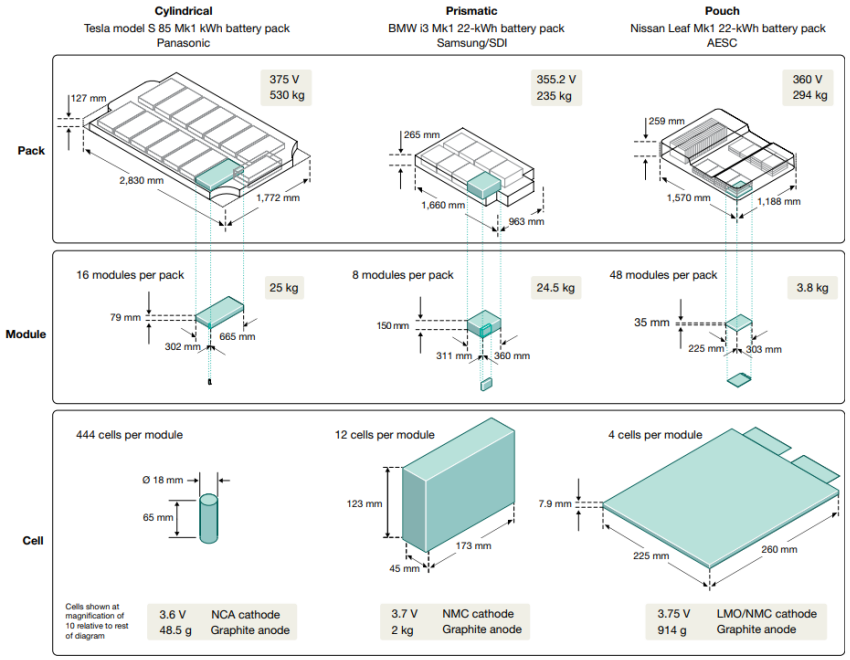


*Figure 2 Examples of three different battery packs and modules (cylindrical, prismatic and pouch cells) in use in EVs [3]*

Within each category of module (cylindrical cell module, prismatic cell module, pouch cell module) there is a great diversity specifications and designs. At present there is no standardisation [6] of design for battery packs, modules, or cells within the automotive sector, and it is unlikely that this will happen in the near future. Other battery-reliant products, such as mobile phones, have seen an exponential proliferation of different sizes, shapes, and types of battery over the past two decades. Much of the factory assembly of these batteries is done by human workers and remains unautomated. Their disassembly and waste-handling typically involve even less structured environments, with much greater uncertainties, than a manufacturing assembly line. This trend will likely continue with the disassembly of LIBs and modules.

Table 1 shows a morphological chart for the combinations of cell configurations, contacts, fixations, and housings commonly found in modules. It is important to understand the enormous variety that this morphological chart implies. Even modules with similar morphological make-ups can exhibit very different designs because subcomponents and their arrangements are design decisions made by the module manufacturer. This makes the development of an automated system capable of handling every design a serious undertaking.

*Table 1 Morphological chart with manifestations of battery module features. Adapted from [7].*

| Component | Properties | Cylindrical | | Prismatic | | Pouch | | |
|---|---|---|---|---|---|---|---|---|
| Cell | Row | Multiple rows | | Single-rowed | Double-rowed | Single-rowed | | |
| | Arrester position | Opposite Side | | Same side | | Same side | | Opposite side |
| Cell Contacting | Connection type | Adhesive bond | | Adhesive bond | Traction/form fit | Adhesive bond | | Traction/form fit |
| | Joining | Welding | | Welding | Screwing | Welding | | Screwing |
| | Bridge | Busbar <with bonding wires> | | Busbar, tab-to-busbar | Hole-busbar, tab-to-busbar | Tab-to-tab | Tab-to-busbar | Screw threads or hole-busbar |
| | Support plate | No | | Yes | No | No | No | No |
| Cell Fixation | Primary fixation (cell-to-cell) | Rigid spacer, spacer strip, mounting bracket | <Gluing> | <Gluing> | | <Cell frame> | | <Gluing> |
| | Secondary fixation (cell-group) | Non-detachable joined side panels | | | Detachable joined end plates and/or side panels <with tension/stretch unit> | | | |
| Module Housing | Function-integration with cell fixation | Yes | No | | | | | |
| | No. parts | N/a | One-part | One-part | Two-part | Two-part | Multipartite | Multipartite |
| | Part joining | N/a | Detachable joined | | | Non-detachable joined | | |

| **Legend** | |
|---|---|
| < > | Optional |

### 2.1.2. Module Disassembly

For repurposing and second-use applications, automotive battery packs and modules are currently dismantled by hand for either the second use of the modules or for recycling. The literature reviewed covers both LIBs and modules for the reason that they both share similarity in disassembly procedures. Disassembly of LIBs and modules is a complex and hazardous task requiring specialised tools and training. The potential hazards found in disassembly include [8] [9]:

- Toxic and carcinogenic materials
- Electrocution of operators
- Short-circuit of the pack
- Rapid discharge, heating, and thermal runway
- Cell combustion and explosion
- Release of noxious by-products

As a result, disassembly takes place in highly monitored and controlled environments, and disassembly procedures are carried out with high precision processes and machinery so as to not trigger an event resulting in one of the aforementioned hazards. Typical procedures involved in module disassembly include:

- Cutting cell connections
- Unscrewing connectors
- Removing wiring, casing, busbars, structural elements, and glue

Research to introduce automation into the disassembly of LIBs has been attempted with varying degrees of success. Using the battery pack of the Audi Q5 as a case study, researchers identified the disassembly routine as a 24-step disassembly sequence across 14 parts, requiring 4 tools, including versatile use of hands [10]. This study highlighted the breadth of tasks, tools, and skills required to fully dismantle a single battery, judging that some tasks are not yet feasible to automate across a range of module designs, such as manipulation of irregularly shaped components.

A follow-up study [11] proposed a design for a robot-assisted disassembly system for the same battery design, concluding that an appropriate degree of automation for the disassembly of EV batteries is currently a hybrid

human-robot system, in which a robot assists a human worker by taking over the task of removing screw and bolt fasteners. A computer vision system was implemented to detect screws and bolts, for the reason that detailed product specifications are generally not available to the recycler, and so it is impractical to assume that exact locations will be available via a database or CAD models.

Another study [12] takes a different approach of using a combination of artificial intelligence and robotics to automate the disassembly of LIBs. With the case study of an Audi A3 battery pack, the research aimed to design a system for automatically generating disassembly plans without precise a priori knowledge of the battery to dismantle. Using a 3D camera, robot arm, Robot Operating System (ROS), a computer vision program and a PC, the system developed is capable of detecting all important components identified in the report (screws, connectors, BMS, modules), and developing a step-by-step plan for disassembly at a high level of abstraction (for example, tasks involve 'remove screws', rather than 'travel to specific location X, Y, descend Z mm and rotate with Q N of torque for 15 revolutions').

Whilst successfully meeting some objectives of the investigation, the technique has only been tested on a single model of EV battery. The author recognised that testing on a larger set of EV battery packs with different components and constructions is necessary, so as to validate the robustness of the proposed method.

## 2.2. Automated Disassembly with Uncertainty

The uncertainties associated with end-of-life (EOL) products have been problematic for human operators and even more so for automation due to the decision making, operating skills, and high level of perception required to disassemble complex and hazardous products such as LIBs. Skilful operators are expected to be capable of accomplishing the disassembly process of previously unseen models of products. Appropriate decisions have to be made based on their prior knowledge and the perceived information regarding the current product's condition.

The task of automated disassembly at scale has so far only been achievable with the use of 'stiff' robotic systems specialised to one product. For example, Apple has implemented an automated disassembly line for the iPhone 6 [13] that can handle 1.2 million phones per year. This line has 22 stations linked on a conveyor system and can take the iPhone apart in 11 seconds. However, this system can only deal with an iPhone 6 model. Intact phones, of this exact model, must be positioned at the start of the disassembly line, which then uses pre-programmed motions of 29 robots to dismantle the phone into 8 discrete parts.

This type of automated disassembly takes place in highly structured environments, in which robots make pre-programmed repetitive actions with respect to exactly known objects in fixed positions. In contrast, the development of robotic systems that can generalize to a variety of objects, and handle uncertainty arising from variability in product condition, remains a major challenge at the frontier of artificial intelligence research. It is important to consider the complexity of module disassembly from this perspective. This demands the use of innovative techniques in the domains of robotic perception, task planning, and use of high-precision actuators and effectors to carry out disassembly routines for a broad range of products, as is the case with battery modules.

Factors that influence automated disassembly

Factors which affect the success of a robotic disassembly system can be categorized into 2 types: external factors and system factors. External factors stem from the EOL products input to the disassembly system. These factors present themselves in the form of uncertainties that a disassembly system may encounter and need to address or manage. System factors are inherent to the disassembly system and are a function of available technology and system design.

*Table 2 External and system factors influencing automated disassembly, adapted from [14]*

| External Factors | System Factors |
|---|---|
| Variation in product structure and condition | Perception capabilities |
| Extent of disassembly required | Cognitive capabilities |
| Quality of outputs required | Tooling and operation capabilities |
| Uncertainty of outcome of disassembly operations | Cost |

Increasing the level of automation in disassembly is currently bottlenecked by lagging development in the critical system factors of perception and cognitive capabilities of robotics. It is clear that advanced sensing technologies in robotic applications such as vision systems are of great importance to the successful automation of disassembly processes [15], and should be a focus of further research and development.

Disassembly Families

Disassembly families are groups of similar-enough products or components that require nearly the same disassembly operations. Optimising a station for a family of components simplifies the design of a flexible disassembly station, at the cost of universal generalisability. For example, in the case of the disassembly case study in [16], printed circuit boards (PCBs) constitute a 'family', allowing a disassembly station to be designed at the expense of other electronic components. By selecting an appropriate disassembly family of components in LIB modules in the research carried out in this report, the challenge of developing a fully automated system is simplified.

Cognitive Robotics

Cognitive robotics is one of the research areas in artificial intelligence that focuses on the problems of reasoning encountered by an autonomous system in an uncertain and dynamic environment.

The use of cognitive robotics is a proposed solution to the challenge of disassembly with product variability. In this approach, a disassembly station has the capability to learn through time how best to perform a disassembly task, where a cognitive robotic agent controls the behaviour of the system in relation to the perception of the environment and existing knowledge [17]. In theory, systems of this design are able to deal with any model of product without supplying specific 'a priori' product information and disassembly processes. As of yet, it remains unclear how feasible it is to develop technology of this sophistication in practice.

Summary

The design of disassembly systems for lithium-ion battery systems from electric vehicles depend mainly on two external factors: the complexity of the battery system design, and the variety of end-of-life battery systems that need to be recycled [18].

What is evident is that the number of permutations for component features across even a single cell variety, in addition to manufacturers' unique design approaches, renders a fully automated system for all module varieties a highly ambitious R&D project.

In the case of automated disassembly of EV batteries, advances in computer vision (CV) and cognitive robotics offer promising tools but this topic remains an open research challenge.

## 2.3. Computer Vision

Research undertaken on CV and methods was conducted following the decision to focus the remainder of this investigation's research on motion planning for the removal of welded connections, explained in section 3.

Robotic sensing is achievable through devices that sense light intensity, sound, proximity, velocity, acceleration, tilt, pressure, temperature, among many other physical phenomena. When fitted into robotic systems, they allow the system to perceive their environment, used to inform decision making. Due to the physical nature of welded connections, emitting no physical signal that could be traced by a sensor other than visual information, the family of techniques opted for in this project was CV.

CV is a broad field that deals with how computers can gain high-level understanding from digital images or videos. It come in many forms, including scene reconstruction, image restoration, motion estimation, pose estimation, image classification, image segmentation and object detection. For the purposes of the vision system proposed in this research, object detection will be most useful, and was investigated in more detail. Object detection (OD) is an active research area in computer vision. Due to occlusion, different viewpoints, variations of illumination and scale in cluttered real-world scenes, the objects of the same class usually exhibit great visual variations, which makes the task challenging.

To conduct further research in the context of identifying welded connections across a variety of module designs, it was considered important to identify OD techniques that worked well in identifying objects in cluttered environments, with varying degrees of lighting, and that were flexible to a range of geometries and sizes of welded connection. In the literature, two OD methods were identified to have been used for components similar to welded connections.

In [11], a Haar cascade [19] was trained to classify screws found in the case study LIB. In evaluating the use of the Haar algorithm to identify whether an image contained screws and bolts, results were found to be unsatisfactory: to be able to detect 50% of the screws, there was over 50% chance for a false positive in this study. The use of a Haar cascade was implemented for the same task on LCD screens in [20] with somewhat improved results: the classifier was able to detect over 80% of screws – however, they also indicated a high number of false-positive screw detections.

Research documented in [21] evaluates the use of a deep learning technique for the detection of cross-recessed screws, a common fastener used in electronics. Version 2 of the YOLO (You Only Look Once) object detection system is used to identify and locate the position of the screws. The work presented in this paper offers a marked improvement over previous methods, with the best-trained model yielding an average precision (AP) of 99.2%.

The research in this section identified two object detection methods used in the literature reviewed, specifically in the context of electronic equipment, connectors, and LIB components: the Haar Cascade Classifier, and a deep learning (DL) OD architecture (specifically of the YOLO family). The accuracy achievable by using a Haar cascade was generally lacklustre, potentially owing to poor implementation, potentially owing to the weakness of the underlying algorithm in context. The best performance was achieved by the implementation of YOLO, and for this reason it was decided to experiment the development of a proof-of-concept motion planner using a DL OD method to act as the motion planner for the disassembly of welded connections.

Training and optimising deep learning systems and training neural networks can be a complex undertaking, and due to a lack of experience on the subject, it was decided to find another, simpler CV method to implement in case it was not possible to generate useful results from a deep learning object detector. For this purpose, the Hough Circle Transform was opted for, based on the hypothesis that the circular faces of cylindrical cells would be

identified using this technique, and that Hough Circle is a standard computer vision method for circle detection [22].

Digital Imagery

In human brains there exists a complex system that interprets visual information with astonishing ease that is as yet little understood. In a machine vision system, however, a computer receives a simple grid of numbers.

For grayscale images, the pixel value is a single number that represents the brightness of the pixel. The most common pixel format is the byte image, where this number is stored as an 8-bit integer giving a range of possible values from 0 to 255. Typically, zero is taken to be black, and 255 is taken to be white. Values in between make up the different shades of grey. To represent colour images, separate red, green, and blue (RGB) grids must be combined – so that each pixel is represented as a vector of intensities of RGB.

This simple format allows a wide variety of mathematical methods to be applied to images, from image compression to feature extraction, image transformation and processing.

Evaluating Object Detectors

When predicting bounding boxes, it is necessary to determine whether a bounding box prediction is good or not based on how well the predicted and actual bounding boxes overlap. This is defined as the area of the intersection divided by the area of the union of a predicted bounding box ($B_p$) and a ground-truth box ($B_{gt}$) and is referred to as "intersection over union" or IoU (equation 1). A perfect bounding box prediction will have an IoU of 1.

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \quad (1)$$

In practice, a positive detection is one in which the IoU exceeds a defined threshold. It is standard to assume a positive prediction of a bounding box if the IoU is greater than 0.5, e.g., they overlap by 50% or more.

The confidence of a prediction is the probability that an object lies within the bounding box, determined by the model. Among other causes, the number of false positives/negatives is dependent on the confidence score threshold chosen. In the application proposed in this research, the consequences of an unreliable model for object detection are severe, where false positives could remove material from unwarranted areas. Therefore, a high enough confidence threshold should be determined so that false positives are minimised at the cost of false negatives.

mAP_0.5:0.95 is a metric combining the true positive, false positive, and false negative counts to provide a holistic measure of an object detector's performance on labelled data and was used to evaluate model performance throughout the research methodology in this report. A full description of how it is calculated is covered in Appendix A.

# 3. Objective Selection

The research summarised in sections 2.1 and 2.2 of the literature review revealed that there is a lot of progress to be made in the perceptive and cognitive capabilities of robotics before a fully autonomous system can be developed to disassemble all types of module.

This is due to:

- Complexity of module designs – modules require dexterity and sensitive feedback to disassemble and can be extremely hazardous.
- Variety of module designs - there is limited standardization resulting in great variance in battery modules.
- Capabilities of traditional robotics - existing automation methods are hard-programmed and specialised to a single design. Robots are not yet able to easily learn how to perform a range of complicated tasks with uncertainty as effectively as trained specialists.

Robots have proved to work well in structured environments. The complexity of the environments and tasks necessitated by module disassembly requires new techniques and technology. Two studies reviewed in the literature were able to automate the disassembly of LIBs to some extent. Wegener [10] [11] implemented a vision system to classify screws on LIBs to moderate success; Choux [12] developed a high-level task planner to identify the main components (modules, screws, empty screws, and cabling) of an A3 Sportback e-tron LIB, and the order in which procedures should be carried out for their disassembly.

## 3.1. Component Selection

To find a task to automate, it was decided to isolate a single component family common across a number of module designs. It has previously been demonstrated that it is feasible to automate the disassembly of screw connectors in some contexts [11] [12] [20] [21]. It was necessary to identify another component that was under-researched, common across module designs, and for which automating disassembly will be of value.

*Figure 3 Two variants of welded connections found in cylindrical cell modules [34]*

Table 1, the morphological chart of components found across module designs, shows that welded connections can be found in cylindrical, prismatic, and pouch cell modules. Although the nature of the connections can vary within and between each cell type, they are one of the more standard features found across modules. In cylindrical cells, they are found along the module faces at the cell terminals (see Figure 3).

Research indicated [23] that it is possible for power tools with precise cutting or abrasive attachments (milling machines, belt sanders) to effectively and safely separate welded connections from cell terminals. Additionally, for a given module there are typically many identical welded connections which can be dismantled by repeated action. This enables the task to be carried out by a simple robotic effector, and since the process is time-consuming and highly repetitive when done manually, this task is well suited for cost-saving through automation.

By focusing on the component of welded connections in cylindrical cell modules, the task of identifying and testing methods that would be used in an autonomous disassembly station was greatly simplified.

## 3.2. Task Selection

Ideally, the robot would solve the entire problem in one go: observations in the form of raw sensor feeds go in, and power to motors used in disassembly routines comes out. In practice it is typical to decouple different aspects of the problem and treat them independently [24].

In general, robotic systems can be viewed as the combination of three integrated subsystems responsible for:

1. Sensing (perceiving the state of the robot and its environment)
2. Effecting (changing the state of the robot and its environment)
3. Processing (taking perceptions, deciding on actions, instigating, and controlling them)

Each subsystem can be broken down into constituent subsystems requiring unique specialisations to design.

Given that the goal of this research is to assist in the development of a fully autonomous system capable of disassembling modules from a broad spectrum of products, a strong ability to perceive its environment will be a fundamental competence of the system. Having identified robotic perception as a key bottleneck factor in the capability of automated systems, and evidence that computer vision could work well as a sensing technology, it was decided to investigate the feasibility of using computer vision to assist in motion planning for the disassembly of welded connections, and to test this across multiple variants of cylindrical cell modules. This research is summarised in section 2.3.

### 3.2.1. Key Assumptions

In order to develop a computer vision system, it's necessary to estimate the design of the disassembly station at a high A simplified schematic of this station might include:

1. A platform to load modules onto
2. A robot arm with an effector to perform the task of milling
3. A 2-D camera to provide the sense data for the robot's operating system
4. Hazard prevention and monitoring equipment

The following key assumptions will have to be made of this hypothetical disassembly station in order to define the requirements of a proof-of-concept system, the modelling exercise of this research:

1. It is possible for welded connection removal to be performed by a CNC milling machine that requires co-ordinate data for motion
2. The location to cut is assumed to be the centre of the cell. Tooling with an appropriate cutting radius will be able to remove welded material located closer to the circumference of the cell.
3. It is possible to control the depth to which the milling piece cuts into the welded connections so as to minimise unnecessary damage to the cells on an individual basis

### 3.2.2. Problem Definition & Proof of Concept Requirements

With these assumptions in place, the remaining challenge becomes to identify the location of welded connections based on the 2-D imagery (input from a camera) and define the path in 2-D that an effector should trace to successfully remove all of the welded connections. A path is a sequence of points in geometric space that the effector will follow, the task of finding a good path is called motion planning.

The central focus of this research can now be defined as developing a proof-of-concept motion planning system for the automated disassembly of cylindrical cell lithium-ion battery modules.

The system should be able to adapt itself to new types of cylindrical cell module, and the variations in welded connection style that come with it. This will most likely require some amount of human intervention to re-configure the system when uncertainty arises. A central objective of this research is to find a technique that minimises the time taken in this intervention. A list of requirements for the proof-of-concept system are specified in Table 3.

*Table 3 Requirements for the proof-of-concept system*

| Index | Requirement | Verification |
|---|---|---|
| 1 | Identify all milling locations within a 3mm radius of the ground truth (defined as the centre of the cell) | Euclidian distance from prediction to ground truth |
| 2 | Make location inferences at least 10x faster than a human labeller | Human time trials |
| 3 | Quickly configurable for new module types – preferably take no time at all to reconfigure | Measure tasks and time taken for reconfiguration process |
| 4 | Use standard hardware and equipment | Equipment and tooling used must be consumer grade |
| 5 | Integrate with robotic operating system and develop motion planning capability | Verify that the output of the proof-of-concept can interface with a robot operating system |

# 4. Relevant Theory

Summary of Methodology

At a very high level, the methodology outlined in Figure 4 sets the context for the topics covered in this section.

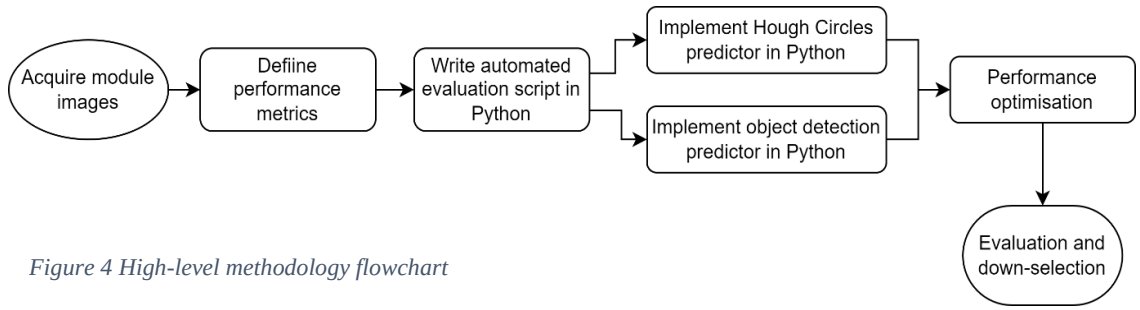


*Figure 4 High-level methodology flowchart*

With a decision made on which component family to disassemble and the disassembly task to automate, the remainder of this report describes the process of developing a proof-of-concept technique to act as a welded connection predictor and motion planner for the task. Research began in computer vision methods, specifically in the domain of object detection and which methods could perform well in the proposed application. This research was summarised in section 2.3 – the outcome of which was to implement two CV algorithms to detect welded connections across multiple module designs: a deep learning object detection model, and the traditional CV method of Hough Circle Transform.

## 4.1. Object Detection

The object detection being referred to in this report is a form of Deep Learning (DL) with Artificial Neural Networks (ANNs). Deep Learning is a subset of techniques found in the field of Machine Learning (ML). Machine Learning is the practice of developing and implementing computer algorithms that can improve automatically through experience the use of data. DL is a specialisation of ML based on artificial neural networks with many layers.

Object Detection is concerned with detecting instances of semantic objects of a specific class (such as humans, buildings, or cars) in digital images and videos. The problem definition of object detection is to determine where objects are located in a given image, i.e., object localisation, and which category each object belongs to – object classification.

The essence of implementing an object detection methodology is to obtain a dataset of typical images with the object that is desired to be detected, to label the objects in the images, and to 'show' an OD model the images so that it can learn to recognise the location and classification of an object when it is presented with an image containing said object.

An advantage of using OD is that the same model could learn to recognise different welded connection types from training data and so can be continually improved. This increases the generalisability and accuracy of the system over time.

Typically, deep learning techniques require large training datasets in the order of hundreds of images and thousands of labelled objects to make effective predictions. Despite this, the field is rapidly advancing, so it is anticipated that future models will need less data to train. Furthermore, since welded connections across a module bear some resemblance to one another, are not occluded in imagery, and are arranged in similar way across images, the task is a relatively straightforward one.

### 4.1.1. Data Formats – Images and Labels

The data input to a deep learning object detector is split into images and their corresponding 'bounding box' labels. The images are represented as a 2-D array of pixel values, and the image content used in training should be representative of what will be submitted to the algorithm during normal use [25].

Labelling is a time-intensive process, requiring human labour to manually draw bounding boxes around objects in an image, and associate a class label with the object. The labels are stored in a data structure separate to the images but are indexed to the image they are related to. There is no single standard format when it comes to image labelling - the data structure they are stored in varies between models.

### 4.1.2. Object Detector Evaluation

Once a model has made predictions on input data, evaluation metrics provide meaningful measures of the model's ability to detect objects.

mAP_0.5:0.95 was chosen as the metric to evaluate the accuracy of bounding box predictions during the model tuning phase of this research. For a full description of how mAP is calculated, refer to Appendix A.

### 4.1.3. Train–Validation–Test Split

To compare how well different models have learned to make predictions in reality, the dataset being used is split into three tranches: training, validation, and testing. This technique aims to evaluate the ability of the models to generalise to unseen data, providing a more accurate understanding of 'real-world' performance.

The training dataset is a sample of data that is used during the training phase - when model parameters are being updated to improve performance. A validation dataset is a sample of data held back from training the model that is used to give an estimate of model skill while tuning model performance (see Model Optimisation). The dataset used to evaluate the final model performance is called the test set. This is data not yet seen by the model during training or tuning and gives the least biased evaluation of model performance.

### 4.1.4. Model Training

Artificial neural networks are at the heart of DL. ANNs are a series of layers of nodes, also known as neurons. In a simple ANN, neurons are connected to every neuron in the previous and next layer. The connections have 'weights' that are learned during training. Data is passed from the input layer to the output layer, where predictions are made. By tuning the weights between nodes, a model's predictions change. The objective of machine learning is to improve these predictions by minimising a loss function through gradient descent. Once a network has been learned, its state can be saved and deployed on computational machinery with sufficient memory and software to make predictions on input data.

The latest DL approaches may achieve high accuracy in many prediction tasks; however, this jump comes at the cost of an increased requirement for processing power. DL requires enormous computing resources for training, and to a lesser extent for inference. It is therefore essential to have dedicated hardware in the form of high-memory GPUs for training [26].

Typically, training a DL model also requires a large amount of labelled data, acquiring and labelling this data can be a time-consuming task. Three approaches to improve training results were implemented in this research.

**Image pre-processing:** it is not always necessary to train on raw image data. In the case of object detection, grayscale images are often used in cases where objects are not sensitive to colour. This speeds up training time because the dataset is reduced by two thirds as a result of 3 arrays of RGB information being reduced to 1 grayscale array. Often in training, images are compressed to speed computation at little cost to accuracy.

**Data augmentation:** model performance always improves when trained with more, representative data. Where it is not feasible to collect more data, the next best thing to real data is half-fake data, this is known as data augmentation. Mosaicking is a form of data augmentation used in object detection, in which labelled images are cropped, then stitched together with crops from other images in the training dataset to generate more training data.

**Transfer learning:** rather than train a model from scratch, transfer learning pre-trains a model on a large dataset which is used as a starting point to train for different but related problem. By applying transfer learning to a new task, it is possible to achieve higher performance than training with only a small amount of data. Even if there is little to zero improvement in performance, it is rarely negatively affected when using a pretrained network [27].

### 4.1.5. Model Optimisation

The simplest way to get better performance is typically to get more representative data to train on. Whilst guaranteeing performance improvements, this task can be time consuming and expensive at scale.

If no more data is going to be collected, the last step in boosting performance is through model tuning. Complex ANN architectures have a large number of hyperparameters. Hyperparameters in ML control various aspects of training and finding optimal values for them can be a challenge. See Appendix B for a list of hyperparameters for the OD model used in this research.

Random search is a simple yet effective methods for finding these values; it works by randomly selecting hyperparameter values over pre-defined distributions. The best performing combinations can then be evaluated

on the test set. Grid search is a similar approach that systematically uses all possible combinations over discrete distributions, and whilst it is exhaustive, it is computationally expensive.

A more advanced method is hyperparameter evolution using a genetic algorithm. Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover, and selection [28].

There are two important hyperparameters of object detector training that are discussed often in this report – batch size and epochs. The batch size is a hyperparameter of gradient descent that controls the number of training samples to work through before the model's weights are updated in each iteration of the backpropagation algorithm. Increasing batch size increases the memory requirement of the GPU to handle the calculations performed during trained and introduces instability but can speed up training. The number of epochs is a hyperparameter of gradient descent that controls the number of complete passes through the training dataset. Increasing the number of epochs increases train time and improves performance on training data up to a point, where accuracy converges.

## 4.2. Feature Extraction

Hough Circle Transform

The Hough Circle Transform (HCT) is an image feature extraction method that works by identifying regions with circular geometrical features. In this case, the feature of interest is the welded connection. In accordance with assumption 2 in section 3.2.1, it can be observed that for all of the modules studied in this research, welded connections are found near the centre of the cell terminals. Therefore, by locating the circular faces of the cell terminals, it is possible to define the area in which the welded connection can be found.

In computer vision and image processing, a feature is a piece of information about the content of an image; typically, about whether a region of the image has certain properties. Features may be specific structures in the image such as points, edges, or objects.

Feature detection includes methods for computing abstractions of image information and making local decisions at every image point whether there is an image feature of a given type at that point or not. The resulting features will be subsets of the image domain, often in the form of isolated points, continuous curves or connected regions. The detected features can then be extracted using a feature extraction algorithm, like the HCT.

A circle's geometry can be described with the following parametric equations, requiring 3 parameters: $x_{centre}$, $y_{centre}$, (defining the circle centroid) and R (defining the circle radius).

$$x = x_{centre} + Rcos(\theta) \tag{2}$$

$$y = y_{centre} + Rcos(\theta) \tag{3}$$

HCT is able to determine these parameters of circles in an image when a number of points that fall on the perimeter are known. The points detected are the output of a Canny edge detector that pre-processes image data before feature extraction in the HCT. The inner working of the algorithm is mathematically involved and is covered in documentation [29].

# 5. Methodology and Experimentation

The methodology of this research was briefly summarised in section 4 to provide context for the underlying theory of techniques applied to the development of a proof-of-concept cell connection identifier. This section expands the level of detail of the methodology, and explains the systematic approach taken.

The majority of modern machine learning and computer vision is accessed using the Python programming language. All code referred to in this research was written in Python 3 using Jupyter Notebooks, version controlled with GitHub, and is accessible at [30]. Jupyter notebooks are documents that can execute python code, and can be identified by their *'.ipynb'* file extensions. These documents are referred to as programs in this research.

The development of a detector was carried out using two unique methods, Hough Circular Transform (HCT), and a Deep Learning Object Detector. Each has different data requirements, and work in different ways. The output would be the same in each case: a set of co-ordinates in the image, defining points to move the cutting tool towards.

Whichever approach resulted as the best performing detector, it was a critical requirement that the technique be able to generalise well across module designs, such that the process of configuring the system to develop motion plans for unseen modules is as automated as possible.

To this aim, four unique module designs were included in this study seen in Figure 5.

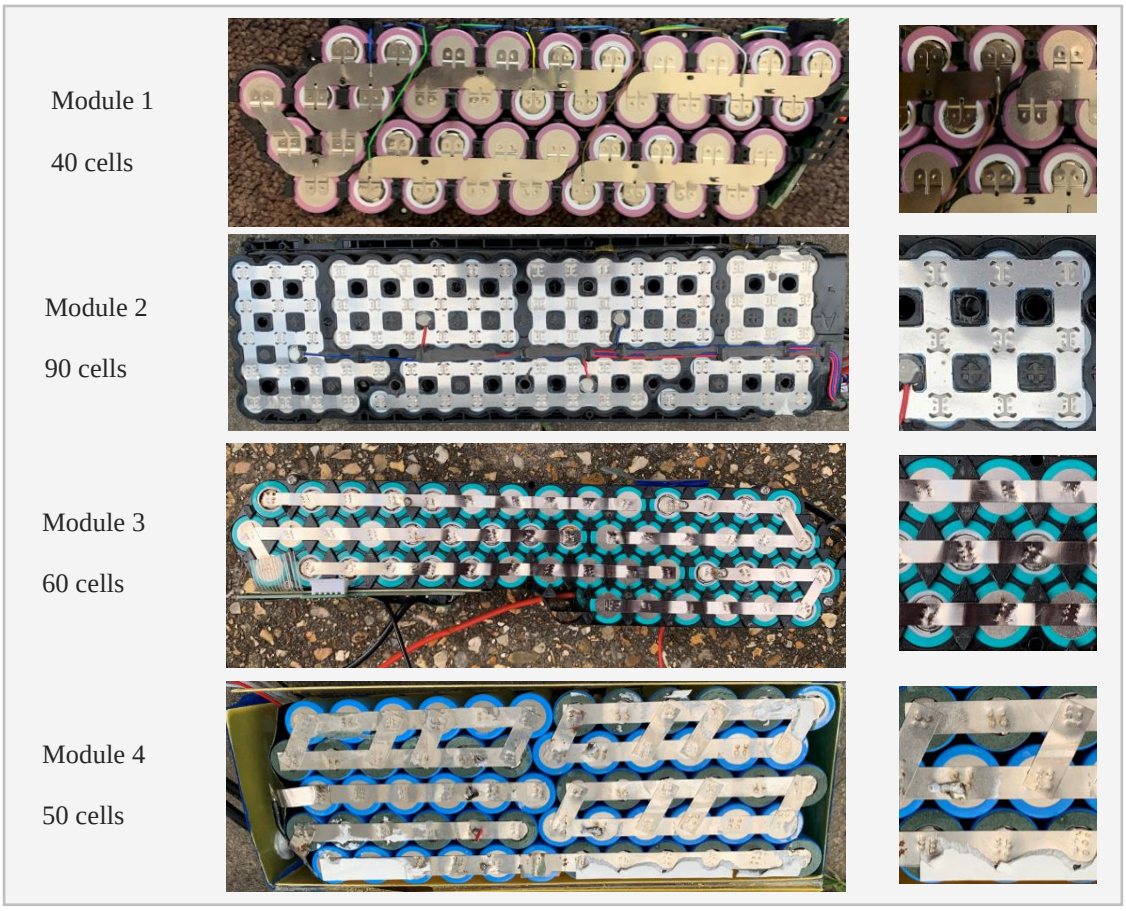


*Figure 5 The four module designs used, and close-ups of their welded connections*

Each module's style of cell connection shared some similarities, but there are some clearly identifiable differences in form. For example, where the tabs in module 1 are neatly welded to the cell in two 'forks', the tabs in module 4 are coarsely welded with significant variety in the number of weld spots on each cell. Furthermore, on modules 3 and 4 some of the cells are occluded by material left over from uncovering the module. This was deliberately done to test the robustness of the detection method to deal with the variation in cell quality that might be seen in an industrial context.

## 5.1. Image acquisition

Images used to develop and evaluate the prediction algorithms should be representative of the images that would be passed to the computer vision system in the application environment. It is assumed that the station camera will point perpendicularly towards the module face – for this reason, images were captured vertically above the modules.

25 images of each module were captured with a 12-megapixel camera (4032 width x 3024 height) at ISO 25 and f1/8. Images are natively captured in .heic file format but were converted to the .jpg format using Python's PIL library.

## 5.2. Evaluation pipeline

The output of the code written to implement each algorithm [30] is a set of x-y co-ordinate pairs designating where to cut. This needs to be compared with a ground-truth of where should actually be cut.

A single test image was selected as the evaluation case for each module. The image was chosen to be the most representative of what would be seen by the vision system in the proposed disassembly station, taken normal to the module face and containing all cells.

Predictor performance was evaluated after every configuration change of the HCT and object detector algorithms, so as to optimise performance by understanding which configurations improve results. This section describes the process of establishing the ground truth that each set of predictions was compared against, and how prediction performance was measured.

### 5.2.1. Ground-truth labelling

The ground truth labels were made to be the centre of the cells – with an assumption that a 3mm deviation from the centre of the cell would lie within the cutting tool's radius and would therefore be of sufficient accuracy to remove all necessary welded material.

To generate a dataset of the ground truth for each module, a program *ground_truth_labelling.ipynb* was written, making us of OpenCV, a Python library, and its click event function to record a list of x-y coordinate pairs that are clicked on through OpenCV's graphical user interface.

The output is stored in two lists, which are saved to .json files in a folder corresponding to the module. These files are referenced in the evaluation program, comparing the values of predictions to the values in these files.

Each module's test image has a unique associated mm/pixel ratio, also known as spatial resolution. This is because the depth of the images was not controlled, so that more diverse images could be taken. This problem is solvable because the diameter of each cell is known to be 18mm, as each cell is an 18650-lithium-ion cell. An online tool [31] was used to measure the pixel distance of a cell's diameter, such that a ratio can be calculated. There is some amount of lens distortion that varies the cell diameter in pixels between cells at the edge of the module against those in the centre, but this is small enough to be ignored.

### 5.2.2. Evaluation Metrics and Automated Evaluation

With a set of predictions output by the methods and the ground truth, the next step is to get a meaningful measure of the method's performance. In the case that there is a prediction for each cell, the most intuitive measure of how well a detector performs is the distance between each prediction and its corresponding ground truth label. In reality, the nature of predictions varies significantly in quality, making evaluation of predictor performance a challenge

To make this clear, three proposed measures of performance are defined, and then a case study described to highlight the challenge.

**Metric 1 (label difference):** Absolute difference between the number of ground truth labels and predictions.

$$\left| n_{labels} - n_{predictions} \right| \tag{4}$$

The best predictor will score 0 in this metric – the number of predictions equals the number of cells. This metric can be misleading in cases where prediction accuracy is poor, but the number of predictions just so happens to be close to the number of labels. Figure 6a shows such a case.

**Metric 2 (distance error):** Sum of errors in mm, calculated as the smallest Euclidian distance between each prediction and its nearest ground truth label

$$mm{:}\,pixel\ ratio\ * \sum_{i=1}^{n_{predictions}} \sqrt{(x_i - \check{x})^2 + (y_i - \check{y})^2} \tag{5}$$

Where $x = \{x1, x2, x3 \ldots x_n\}$, $y = \{y1, y2, y3 \ldots y_n\}$, corresponding to the x-y coordinate pairs for the n predictions, $x_i$ is the x-coordinate value for the $i^{th}$ prediction (similarly for $y_i$), and $\check{x}$ is the nearest ground truth label to $x_i$ (similarly for $\check{y}$ to $y_i$).

A smaller score on this metric generally indicates better performance. Unfortunately, a low number of predictions will typically score well in this metric – meaning that a good policy for a predictor to optimise for this metric would be to make few or no predictions at all, as seen in Figure 6b.

**Metric 3 (missed labels):** Number of ground truth labels that do not have a prediction within a 3mm radius. Note that as an additional measure, the limit was set to a 1.5mm radius, to provide a comparison of model performance.

It should be made clear that this is not equal to the number of false positives (the number of predictions made that do not correspond to a ground truth label) – it considers the ground truth labels and whether a passing prediction has been made for it. This metric's weakness lies in situations where a predictor makes a large number of predictions and enough of them fall within the threshold to qualify as true positives - what is not recognised is the potentially thousands of erroneous predictions made. This scenario is shown in Figure 6c.

Case Study

To illustrate these metrics and the cases in which they independently fail, consider the following 3 hypothetical predictor outputs for module 2. Module 2 has 90 cells, and therefore 90 ground truth labels.

*Table 4 Case study of 3 hypothetical predictors*

| Predictor | Predictions | Features |
|---|---|---|
| 1 | 90 | Some predictions pass as true positives, but some cells have duplicate predictions, and some predictions are far from cells. This predictor scores well on metric 1, but poorly on metrics 2 and 3. |
| 2 | 9 | Each prediction passes as a true positive, but many cells have no predictions. This predictor scores well on metric 2, but poorly on metrics 1 and 3. |
| 3 | 900 | Every ground truth label has a passable prediction, but most predictions are duplicates or far from cells. This predictor scores well on metric 3, but poorly on metrics 1 and 2. |

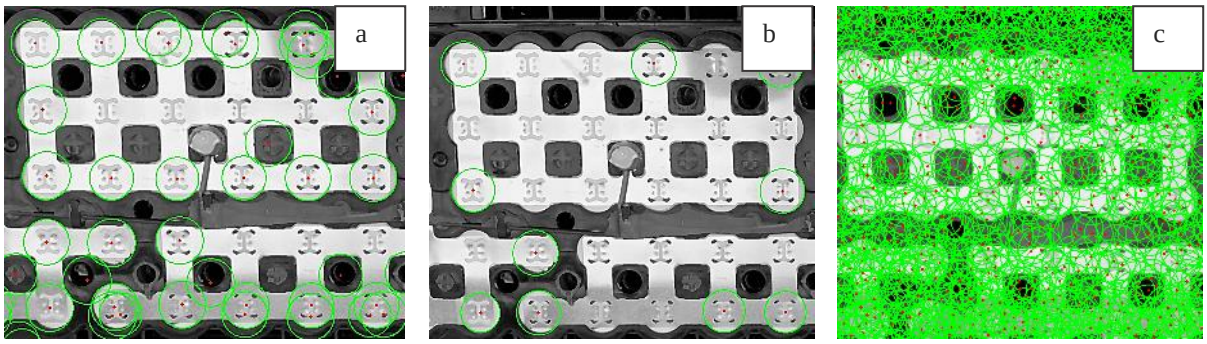


*Figure 6 Visualisation of hypothetical predictors 1(a), 2(b), and 3(c). Generated using Hough Circle Transform*

A 'perfect' set of predictions scores 0 on both the label difference and missed labels metrics. Without perfect predictions, performance was optimised by carefully balancing the trade-off between using all 3 metrics to assess accuracy, as well as visualisations of predictions to evaluate a predictor's ability.

The program *evaluation.ipynb* was written to calculate the three metrics defined above as well as provide useful visualisations of the predictions, seen in Appendix C.

## 5.3. Method 1: Hough Circle Transform

Two Python libraries were found to have developed implementations of the Hough Circle Transform (HCT), scikit-image, and OpenCV. OpenCV was opted for in this case because it is an industry standard for computer vision and has the more in-depth documentation.

The HoughCircles function in OpenCV has an internal Canny edge detector (described in relevant theory). It takes grayscale images as input and outputs a vector of 3 values ($x_{centre}$, $y_{centre}$, and R) for each circle detected. 5 key arguments that influence the performance of this method are listed in Table 5:

*Table 5 List of key parameters passed to OpenCV's HoughCircles function*

| Argument ID | Definition | Default Value |
|---|---|---|
| minD | Minimum distance between detected centres. | 16 pixels |
| p2 | Upper threshold for the internal Canny edge detector. | 100 |
| p1 | Threshold for centre detection. | 20 |
| minR | Minimum radius to be detected. | 0 – circles of all radii will be detected |
| maxR | Maximum radius to be detected. | 0 – circles of all radii will be detected |

The influence of the minD, minR, and maxR arguments on prediction results is interpretable from the descriptions. Assuming the cell size is moderately regular, for a given image, there is a range of values that cell radii take. Setting the minR and maxR arguments to this range will ensure the possibility of the cells being detected. Setting minD correctly prevents duplicate predictions being made for the same cell, but at too high a value will reduce the number of detectable cells.

The influence of p1 and p2 on the ability of the HoughCircles algorithm to make accurate predictions is not so obvious. Configuring these parameters to optimise performance of the algorithm to detect cells across all 4 modules was the main challenge of this approach.

The method developed for this research, written in *optimise_hough.ipynb*, works by iteratively searching through the parameter space (combinations of values of p1 and p2 within defined ranges) and logging performance using the three metrics developed for this application. With this data, it is possible to tune parameter combinations and identify those that perform well for a given image. It was hypothesised that the most performant parameters for predicting cell faces on each module image would be approximately the same value.

What follows is a description of an example manual tuning process for module 2. Firstly, the three arguments minR, and maxR were found through manually searching values and visually inspecting predictions at each set of combinations. Manual tuning was performed by 'sliding' a window of minR and maxR values down from large pixel values. This was done because small pixel values of minR result in a large number of smaller circles being detected in the image, which was computationally expensive and time consuming. Through a similar process of manual tuning and visual inspection, an appropriate value for the minD argument was established.

Next, p1 and p2 were found using the iterative method. Many combinations were not able to make any predictions at all, returning an error for the HoughCircles function. Those that successfully made predictions were evaluated against metrics 1, 2 and 3, and their scores saved.

Contour plots for each metric were generated using matplotlib, a Python plotting library. This was used to gain and understanding of the complex relationships that the parameter values have on each metric of performance. The visualisations of Figure 7 (a-c) show a clear trade-off between metrics 2 and 3 (this was true for all modules), and two regions where metric 1 (label difference) is minimised, around (94, 21) and (100, 17) in this case.

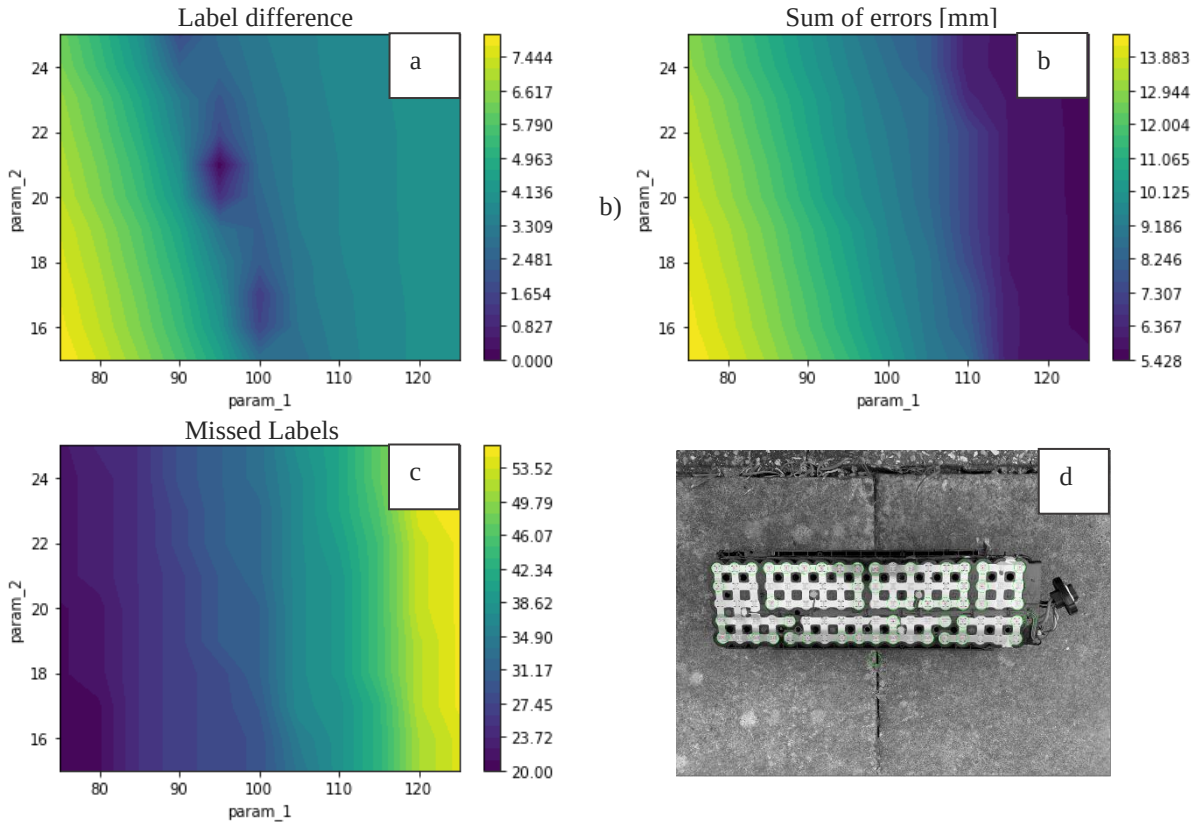


*Figure 7 a-c) Contour plots of module 2's final param_1 (p1) and param_2 (p2) search process*
*d) Image used during visual inspection*

With a more intuitive grasp of the data, it is then possible to comb through more promising ranges of combinations: each set of iterations trialled 100 pairs of p1 and p2.
The results can then be searched over to return parameters that meet defined metric conditions. The following pseudocode illustrates this method (note that in this pseudocode, the metric condition thresholds are arbitrary):

**FOR** result in results
**IF** (metric 1 < 40) **AND** (metric 2 < 60) **AND** (metric 3 < 10):
    **RETURN** p1, p2

The conditions can be made progressively more restrictive until only a handful of candidate parameter combinations responsible for the best predictions remain. The predictions of these candidates are visually inspected as a final check, and the best performing set of predictions from this exercise was selected as the parameter combination for that module. Note that in Figure 7d, all predicted circles lie within the module's area – which is not easily discernible from metrics alone, and so requires visualisation to validate

The best performing parameter combination for each module was then used to make predictions for the rest of the modules, to investigate if it was possible to find a single combination of parameters that generalises well to different module designs.

## 5.4. Method 2: Object Detection

### 5.4.1. Model Selection

Modern object detection with deep learning can be divided into two paradigms: one-stage and two-stage object detection. Two-stage detectors are generally associated with higher accuracies and inference times. Models with this approach include R-CNN (Regional Convolutional Neural Networks), Mask R-CNNs, and Fast R-CNNs. One-stage detectors, such as YOLO and Single-Shot Detector (SSD), exhibit a faster inference time, usually at the cost of prediction accuracy. Table 6 shows a comparison between mAP, training frames/images per second (FPS) rate and inference time for each model based on the Voc 07 test set [32].

*Table 6 Comparison of OD model performance on the Voc 07 test set*

| Model | mAP (%) | Rate (FPS) | Inference Time (s/img) |
|---|---|---|---|
| R-CNN | 66.0 | 5.00 | 32.84 |
| Mask R-CNN | 78.2 | 0.03 | 30.00 |
| Fast R-CNN | 73.2 | 0.40 | 0.11 |
| YOLO | 63.4 | 45.00 | 0.02 |
| SSD | 74.3 | 46.00 | 0.05 |

When selecting a model, it was important to first consider those that could be quickly implemented, trained, and iteratively experimented upon to optimise performance. If the accuracy of these models was not satisfactory, then alternatives would have to be considered. One OD system that meets these requirements is YOLOv5. YOLOv5 is the fifth iteration of YOLO architectures and is an active open-source project with development led by Ultralytics.

The YOLOv5 project has made object detection accessible, bypassing the requirements of expertise in building OD models to build a custom object detector. Its strengths are its simplicity to implement, optimise and its speed of training and inference. Furthermore, YOLOv5 is integrated with Weights & Biases – a model training data platform that automatically logs important metrics, including validation dataset mAP_0.5:0.95, and RoboFlow – an image data and labelling management platform. YOLOv5s (small) is one of the model architectures developed as part of the YOLOv5 project and was opted for as the object detection model in this application due to its small scale and fast training times. If predictions were not adequate, larger models could be trained, including YOLOv5m (medium) and YOLOv5l (large). Model hyperparameters are listed in Appendix B.

### 5.4.2. Experimentation Methodology

The central focus of the object detector experimentation methodology was to train YOLOv5s with different model and dataset configurations, in order to identify those that yield the best results with respect to configuration time, training time and prediction accuracy. Prediction accuracy results from each experiment's configuration were generated with *evaluation.ipynb* and stored in an excel spreadsheet. Training data was stored on the Weights & Biases platform.

Training Configuration Experiments

When configuring a training run, there are a large number of factors that will influence how a model trains and performs in its task. These factors come under the domains of data quantity and quality, model hyperparameter settings, hardware used, batch size and epochs.

When experimenting with a configuration factor, all other factors were kept the same in order to isolate the influence of the variable factor in the results. The number of epochs trained over remained consistent throughout at 150 epochs. The range of factors experimented with are described out in Table 7.

*Table 7 Configuration factors experimented with in the research methodology*

| ID | Description | Justification |
|---|---|---|
| 1. | Influence of mosaicking: the model was trained with and without mosaic data. | Mosaicking training data could reduce the amount of data that needs to be generated, reducing costs if the model needs re-training to recognise new module designs. It is useful to understand its effect on train time and prediction accuracy. |
| 2. | Influence of transfer learning: the model was trained from pre-trained weights, and from scratch (pretrained on the COCO [33] dataset). | Better performance may be gained from training with pre-trained model weights. |
| 3. | Influence of batch size: the model was trained with batch sizes of 4, 12 and 36. | Increasing batch size speeds up training but can result in higher memory requirements, instability resulting in poor performance. Finding a suitable value will optimise train time at no cost in accuracy. |
| 4. | Influence of image dimensions: the model was trained with a dataset comprised of images compressed to 640x640, and 1280x1280. | Compressing data before training reduces storage requirements, and speeds training. |
| 5. | Influence of dataset size: the model was trained with 25, 20, 15, 10 and 5 labelled images. | It is of interest to understand the smallest size of dataset that can yield results. The smaller the dataset requirement, the easier it would be to train to recognise new modules. |

Generalisability Experiment

The final experiments aimed to understand how well a model trained on images of module 1, 2, and 3 could make predictions on the fourth, and to try and optimise this performance.

The baseline model configuration for this experiment was established using the configurations that yielded the best results of experiments 1 to 5, and the training and validation set was comprised of images from modules 1 to 3. Predictions made on module 4 were evaluated using the 3 metrics, and two hyperparameter optimisation techniques were applied in an effort to improve performance. After yielding no tangible improvements in prediction accuracy through these optimisation methods, the hypothesis that the model was overfitting to training

data was tested by training the same model configuration repeatedly but varying the number of epochs trained over.

### 5.4.3. Model Training and Inference

YOLOv5 is an active project, regularly updated on GitHub. GitHub provides internet hosting for software development and version control using Git. Once downloaded, the YOLOv5 GitHub repository contains all model data and software required to train models and infer predictions with them.

To instantiate training of the deep neural network the program *train.py* is executed. This program has a number of arguments that get passed to it and influence how the training is executed. The most important are in Appendix D. The output is a file that describes the trained weights of the model – and could be uploaded to the machine running the disassembly station's computer vision system.

The program *detect.py* takes a trained model and test images as input, and outputs prediction data. This is the program that would be run when deployed in a disassembly station. Similar to *train.py*, *detect.py* has a number of arguments that get passed to it and influence how the inference is executed. The most important are in Appendix F. Helpfully, passing the argument --save-txt generates a text file with the bounding box annotation data. From the data, the central co-ordinate pair of each bounding box can be extracted – this is what is used as the prediction. This program was implemented as a subroutine of the program *'predictions.ipynb'*, written for this research.

### 5.4.4. Dataset Generation and Exporting

*train.py* requires a specific input data structure and format (Appendix E) and generating it can be a time-consuming task. To speed up the process, labelling software such as MakeSense.AI, LabelImg, VGG image annotator, LabelMe, Scalable, RectLabel, and OpenCV simplify labelling and annotation.

As an affiliate of the YOLOv5 project, RoboFlow offers a wide range of tools that simplifies dataset generation and management including: an annotation interface; image pre-processing and augmentation; train-validation-test split creation; dataset storage and versioning; dataset conversion YOLOv5 structure and format; and an application programming interface (API) integrated with YOLOv5 for dataset export and download. RoboFlow was opted for as the labelling software tool for this research due to its functionality and integration with YOLOv5.

Cells were labelled with a consistent methodology – ensuring that each bounding box edge was tangential to the cell's outer diameter, enclosing the entire cell face.

### 5.4.5. Image Pre-Processing and Data Augmentation

These three procedures were applied to the data through RoboFlow.

**Resize:** Other than when investigating the influence of image dimensions on model performance (experiment 4), **i**mages were resized from 4032x3024 to fit a 640x640 pixel resolution through RoboFlow. The reasons for this are twofold:

1. Training should be at the same resolution as inference. Standard type cameras used in machine vision are 410,000 pixels (640x640) [27]. If a model is to be deployed in industry, it should generally be trained at 640x640. Any higher resolution camera's images can be compressed to 640x640 for inference.
2. Training at a smaller resolution significantly reduces computational intensity of training and train time

**Grayscale:** Given the context of the problem, it was anticipated that including colour information was not going to negatively affect model performance. The main difference between welded connections and the rest of the

image are object edges, highlights, and texture variations that signify the location of a welded connection. Using 3-channel RGB images will unnecessarily use memory and increase training times.

**Mosaic:** Mosaicking is an effective method of generating 'half-fake' data. For a training set of 20 images, RoboFlow allows generation of 3x mosaiced images, increasing the training set to 60 when mosaicking is applied.

Each dataset was split into 80% training examples, 20% validation examples, and 1 test image per module. Testing on one representative image is all that's necessary to validate that the model would work in the application context.

### 5.4.6. Training Methodology and Optimisation

Training machine learning models is computationally intensive, necessitating the use of specialised equipment. GPUs are optimised for the specific operations required in image processing, gradient descent and backpropagation algorithms used when training deep neural networks for object detection. High-performance GPUs are expensive hardware, so access to a physical GPU was not possible within the constraints of this project. Colab is a platform operated by Google that allows free access to high power GPUs (Nvidia Tesla P100) and was used for all training, running the program *train_yolov5.ipynb* on Colab with *train.py* as a subroutine.

Only the module 1 dataset was used in experiments 1 to 5. This was because each module dataset has been labelled the same way, has identical image quality, and has the same number of images. This means that training tasks are similar between modules 1 and 4, such that repeating the experiments for each module would yield similar results.

Experiments 1 to 5 were straightforward to set-up and run – *train.py* was executed once for each configuration, model weights were saved locally and used to infer on the test image, then predictions were saved and evaluated against ground truth.

Testing the ability of the model to generalise to the unseen module 4 was more involved. Results from inference were unsurprisingly not as accurate for module 4 as they were for modules 1-3. Training was already using all of the available data, so it was decided to investigate whether performance could be improved through Hyperparameter Search or Evolution.

<u>Hyperparameter Optimisation: Random Search</u>

A random search hyperparameter sweep was carried out to find sets of hyperparameters that would improve validation dataset mAP_0.5:0.95, with the hypothesis that an improvement in this metric, analogous to model accuracy, would be correlated with improved results when making predictions on the unseen module 4.

The sweep was configured using Weights and Biases, and run data (Figure 8) was saved to the platform. A full description of the hyperparameter ranges passed to the sweep program is listed in Appendix B.

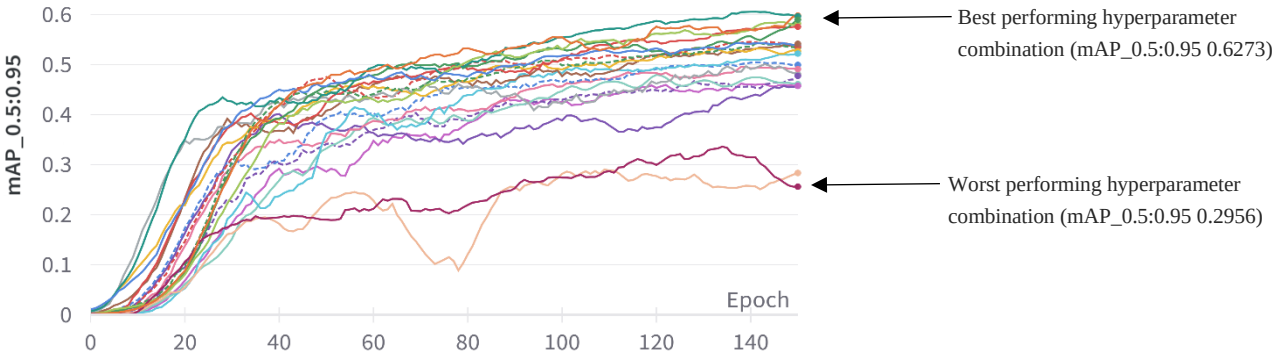


*Figure 8 Smoothed plot of 20 training runs with hyperparameters defined by random search*

It was clear that by randomly searching over hyperparameter combinations, there was an improvement in the best mAP_0.5:0.95 score achieved, increasing from 0.5758 to 0.6273. The best performing model from this sweep was selected and inferred predictions from the test image. As a control, predictions were made by the average and worst performing models from the sweep, and their performance evaluated by metrics 1, 2, 3.

Hyperparameter Optimisation: Evolution with Genetic Algorithm

Evolution is performed about a base scenario which is improved upon. Better initial guesses will produce better final results, so hyperparameters were initialised to those that were used in the model that had yielded this highest mAP_0.5:0.95 up to the time of this optimisation procedure.

Fitness is the value the algorithm seeks maximize and was defined as the weighted combination of two metrics: mAP_0.5 contributes 10% of the weight and mAP_0.5:0.95 contributes the remaining 90%.

Evolution is carried out by passing the --evolve argument to *train.py*, specifying the number of generations to evolve over. Each generation was a full training run of 150 epochs. Over 15 generations, the model saw no significant improvement in performance. Nonetheless, predictions were made with the best, average, and worst performing models from this process, and results recorded.

# 6. Experimental Results

This section summarises the key results of the experiments carried out in the process of developing a proof-of-concept welded connection predictor. Beyond reporting on the ability of both methods developed to make accurate and complete sets of predictions defined by metric 1: label difference (M1), metric 2: distance error (M2), and metric 3: missed labels (M3), data collected on validation set mAP_0.5:0.95, dataset development time, and inference time for YOLOv5s is presented and evaluated in the discussion.

## 6.1. Method 1: Hough Circle Transform

To establish a baseline, the Hough Circles Transform was run on each module with default values for parameters p1 and p2 of 100 and 20, respectively.

Following manual hand-tuning on an individual module basis, predictions improved significantly across all module designs (Table 8). Parameter combinations responsible for these results are summarised in Appendix H.

*Table 8 HoughCircles performance pre and post manual tuning of parameter 1 and 2*

| Module | With default parameters | | | With tuned parameters | | |
|---|---|---|---|---|---|---|
| | M1 | M2 | M3 | M1 | M2 | M3 |
| 1 | 16 | 267.1 | 2 | 3 | 20.8 | 3 |
| 2 | 14 | 1339 | 31 | 41 | 73.9 | 39 |
| 3 | 413 | 25420 | 0 | 3 | 53.1 | 2 |
| 4 | 25 | 164.1 | 0 | 0 | 41.4 | 0 |

### 6.1.1. Generalisability Test

The tuned parameter combinations identified for each module were cross-validated on the other three modules to evaluate whether the best performing parameter combination for one module could work well for the rest. These results are summarised in Table 9. Blue borders indicate the best performing parameter combination for a module. Unsurprisingly, these correspond to the module that they were tuned on. The green (good) – red (poor) colour scheme is scaled to each module's set of results (each row) independently of other module's results.

*Table 9 Cross-validation of tuned HoughCircles parameters.*

| Module | Module 1 Params | | | Module 2 Params | | | Module 3 Params | | | Module 4 Params | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | M1 | M2 | M3 | M1 | M2 | M3 | M1 | M2 | M3 | M1 | M2 | M3 |
| 1 | 3 | 20.76 | 4 | 5 | 19.28 | 5 | 17 | 14.41 | 17 | 11 | 17.82 | 11 |
| 2 | 32 | 165.6 | 34 | 39 | 73.86 | 41 | 71 | 14.2 | 71 | 63 | 33.54 | 64 |
| 3 | 25 | 1739 | 0 | 39 | 2398 | 0 | 2 | 53.05 | 3 | 3 | 167.4 | 1 |
| 4 | 0 | 75.13 | 4 | 0 | 74.98 | 4 | 3 | 41.42 | 3 | 0 | 41.4 | 0 |

## 6.2. Method 2: Object Detection

To build the dataset used for YOLOv5s' training and evaluation, a total of 6000 cells were labelled, taking approximately eight and a half hours in total. Timings taken to label 5 and 10 images are recorded in Table 10.

*Table 10 Dataset generation process summary and timings*

| Module | Cells | Time for 5 images | Time for 10 images |
|---|---|---|---|
| 1 | 40 | 14m 58s | 27m 32s |
| 2 | 90 | 45m 46s | 84m 07s |
| 3 | 50 | 17m 19s | 36m 48s |
| 4 | 60 | 24m 42s | 43m 33s |

### 6.2.1. Experiments 1-5

Unless otherwise specified in Table 11, the following experiments were carried out on the module 1 dataset with default configuration as follows. The dataset was comprised of 25 labelled images, grey scaled and resized to 640x640. The data was then separated into an 80%/20% train-validation split (20 training images, 5 validation images), and the training data was mosaicked to generate 60 training images in total. YOLOv5s was trained from a COCO dataset pre-trained checkpoint over 150 epochs with a batch size of 12. See Table 11 for results.

**Mosaicking** minimally improved mAP, at the cost of an increased training time of 1 minute. Although this is equivalent to a 20% increase in this context, this would only be a problem if a model were being trained over many hours or days and so its use was continued in future training runs. Both were able to make predictions on the test image with perfect accuracy at the 3mm threshold.

**Transfer Learning** offered no significant improvement in prediction accuracy but did not significantly affect any other metrics of performance either. In general, it is standard practice to use pre-trained weights where there is ambiguity on performance improvements [27], and so pre-trained weights were used for future training runs. Both were also able to make predictions on the test image with perfect accuracy at the 3mm threshold.

**Batch Size** did not significantly boost accuracy – for models trained on batches of 4, 12 and 36, each made perfect predictions at the 3mm threshold. As expected from theory explained in section 4.1.5, training times did improve significantly: training with a batch size of 36 almost halved train time. Two pieces of data not included in the table are the increased GPU memory requirement of training with larger batch size, and the instability of the runs with higher batch sizes. When experimenting with much higher batch sizes, results would occasionally drop to 0 mAP and not recover for the remainder of the run.

**Image Dimensions** tested in this experiment had little effect on the accuracy of predictions. Compressing images to 640x640 significantly improves training time, halving train time of runs at 1280x1280, whilst scoring the same accuracy at the 3mm threshold.

**Dataset Size** also had little effect on accuracy at 3mm threshold with the sizes selected, other than the smallest dataset size of 5 labelled images: at a 4-1 train split. The reason as to why performance so steeply degraded some point between 10 and 5 labelled images is an area for further investigation. Although train time between the range of datasets tested does not vary significantly – the time taken to generate the datasets does (Table 10). This is important if this system were deployed in a commercial setting; knowing that 10 images is sufficient to train the model to recognise new representations will save time and costs otherwise spent collecting, labelling, and managing data.

*Table 11 Results from experiments 1-5*

| ID | Variable Factor | Value | Train (min) | Best mAP_0.5:0.95 | M1 | M2 | M3* | M3** |
|---|---|---|---|---|---|---|---|---|
| 1. | Mosaicking | Without mosaic | 5.0 | 0.499 | 0 | 32.9 | 0 | 1 |
| | | With mosaic | 6.1 | 0.538 | 0 | 27.9 | 0 | 1 |
| 2. | Transfer Learning | Pre-trained weights | 5.4 | 0.547 | 0 | 24.3 | 0 | 1 |
| | | From scratch | 5.3 | 0.525 | 0 | 20.3 | 0 | 0 |
| 3. | Batch size | 4 | 8.6 | 0.544 | 0 | 26.2 | 0 | 0 |
| | | 12 | 6.1 | 0.538 | 0 | 27.9 | 0 | 1 |
| | | 36 | 4.6 | 0.578 | 0 | 23.5 | 0 | 0 |
| 4. | Image Dimensions | 640x640 | 6.1 | 0.538 | 0 | 27.9 | 0 | 1 |
| | | 1280x1280 | 12.4 | 0.632 | 0 | 24.1 | 0 | 0 |
| 5. | Labelled Images (Dataset Size) | 20 | 5.6 | 0.585 | 0 | 27.9 | 0 | 1 |
| | | 15 | 5.3 | 0.511 | 0 | 33.9 | 0 | 3 |
| | | 10 | 4.7 | 0.531 | 0 | 79.1 | 0 | 0 |
| | | 5 | 3.7 | 0.026 | No predictions made | | | |

*threshold at 3mm **threshold at 1.5mm

To verify that this level of performance was not unique to module 1, the results of training a single model on all 4 datasets – totalling 240 training images and 20 validation images are summarised in Appendix I.

### 6.2.2. Generalisability Test

The YOLOv5s model results on a module-by-module basis were highly accurate, with all tests passing the true positive criterion of making all predictions within a 3mm radius of each ground truth label, and no under or over-predictions on the count of cells present in an image. The true test of a computer vision system in this context, however, is its ability to recognise welded connections on designs that it has not yet been trained on, as per requirement 3 from Table 3.

Predictions made on module 4 by a model trained on a dataset of modules 1-3 were not of the same standard as the model trained on all datasets. The following baseline (Table 12) was established with data pre-processing, augmentation, and training configuration informed from experiments 1-5.

*Table 12 Generalisability test baseline results*

| Batch Size | Epochs | mAP_0.5:0.95 | M1 | M2 | M3 (@ 3mm) | M3 (@ 1.5mm) |
|---|---|---|---|---|---|---|
| 12 | 150 | 0.606 | 1 | 70.5 | 3 | 19 |

Two hyperparameter optimisation methods were explored to improve on the baseline scores. Models with best, average, and worst mAP_0.5:0.95 scores were evaluated on module 4, and results summarised in Table 13.

*Table 13 Results from hyperparameter optimisation through Random Search and Genetic Algorithm*

| Method | Rank | mAP_0.5:0.95 | M1 | M2 | M3 (@ 3mm) |
|---|---|---|---|---|---|
| Random Search | Best | 0.6273 | 22 | 35.5 | 22 |
| | Average | 0.5185 | 31 | 27.1 | 31 |
| | Worst | 0.2956 | 40 | 19.81 | 40 |
| Genetic Algorithm | Best | 0.5899 | 18 | 31.2 | 18 |
| | Average | 0.5778 | 16 | 30.4 | 17 |
| | Worst | 0.5404 | 17 | 31.3 | 17 |

# 7. Discussion

## 7.1. Proof of Concept: Results Analysis and Verification

### 7.1.1. Method 1: Hough Circle Transform

Using the Hough Circle Transform to locate of welded connections yielded unusable predictions for the proposed application – after manually searching the algorithm's parameter space to optimise predictions for each module, it was not possible to find a combination that could effectively identify all cell connections in an image other than for module 4. Furthermore, performance improvements gained on each module through hand tuning resulted in entirely different parameter combinations (see Appendix H): a set of parameters that generated the best predictions for one module design was rarely transferrable to any other module.

### 7.1.2. Method 2: Deep Learning

YOLOv5 significantly outperformed the Hough Circle Transform method. On a module-by-module basis, models tested on the same module designs as they were trained on always made perfect predictions, when trained with the default configuration described in section 6.2.1.

Initially, predictions made in the generalisability test in 6.2.2 did not meet performance requirements. It was hypothesised that by improving mAP on the validation set, improvements in predictions on the unseen module would follow.

Results from optimising hyperparameters were worse than expected. Even though best mAP_0.5:0.95 had increased, metrics 1, 2, and 3 recorded lower scores for predictions on the unseen module 4.

Training over more epochs, mAP_0.5:0.95 improved significantly, whilst predictions made on module 4 degraded appreciably, as seen in Table 14.

*Table 14 Results of training over 1200 epochs*

| Train | Val | Batch Size | Epochs | Time (min) | mAP_0.5:0.95 | M2 | M1 | M3 (3mm) | M3 (1.5mm) |
|---|---|---|---|---|---|---|---|---|---|
| 120 | 40 | 6 | 1200 | 82.75 | 0.7167 | 496.4 | 16 | 27 | 34 |

This apparent paradox of evaluation metrics was likely a case of model overfitting – a situation in which a model

over-specialises to predicting object representations contained the training data. In this case, the model had improved its ability to recognise cells from images of modules 1-3, at the cost of the ability to abstract its definition of a cell to recognise them in the image of module 4.

By training the same model configuration over a range of epoch values, recording mAP_0.5:0.95 for each model, then making predictions on module 4 and recording the number of missed labels at a threshold of 3mm (metric 3): mAP_0.5:0.95 was plotted against metric 3 for each model trained in Figure 9a. The best fit curve for this plot is similar in form to the literature on underfitting-overfitting trade-offs Figure 9b. In this case mAP is analogous to model complexity. This is because training over higher epochs results in higher mAP, and higher mAP is achieved by a more complex set of weights.

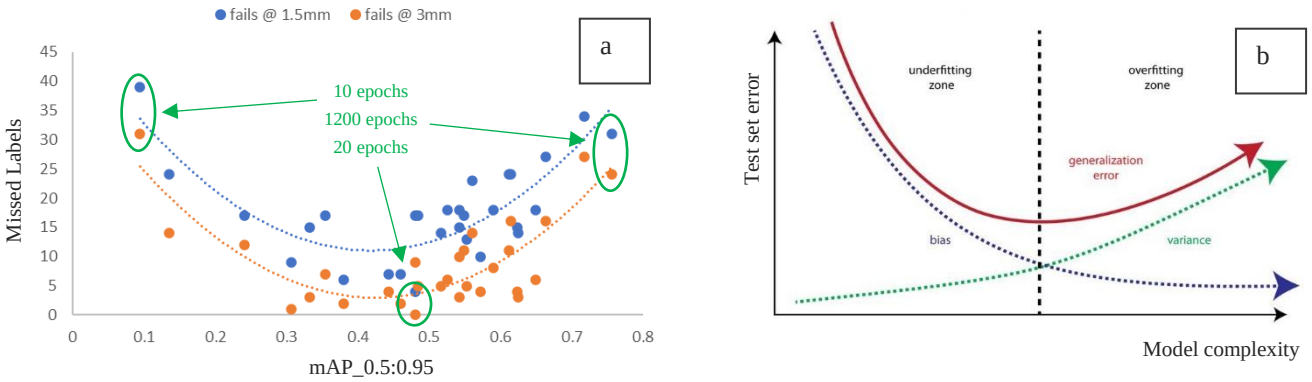


*Figure 9 a) Plot of mAP against missed label error (metric 3) recorded for the same model trained over varying epochs b) Visualisation of the bias-variance trade-off [35]*

At low epochs, the model scored poorly on both validation set mAP and test image predictions – the model underfit the training data and validation (train/val) data and so was not able to make accurate predictions on either train/val data or test data. At high epochs, the model scored well on train/val set mAP, but test image prediction performance was also poor.

The best performance achieved by the models' predictions on module 4 occurred between mAP_0.5:0.95 values of 0.3 to 0.5, with perfect predictions made by a model achieving mAP of 0.48. The same model was evaluated on the 3 models it was trained on and made perfect predictions for those as well, as seen in Table 15. In this table, 'Inference' is the time taken for YOLOv5s to infer predictions with *detect.py*. 'Time for Results' is defined as the total execution time of code written to extract co-ordinate information and log it to .json files. If implemented in a disassembly station, code quality should be optimised to significantly shorten this time.

*Table 15 Final results of predictions made by highest performing model trained on modules 1, 2 and 3*

| Module | Inference [ms] | Time for Results [s] | M1 | M2 | M3 (@3mm) | M3 (@1.5mm) |
|---|---|---|---|---|---|---|
| 1 | 140.5 | 5.88 | 0 | 22.8 | 0 | 0 |
| 2 | 403.1 | 5.91 | 0 | 49.7 | 0 | 0 |
| 3 | 306.1 | 5.81 | 0 | 38.3 | 0 | 1 |
| 4 | 250.1 | 5.96 | 0 | 64.4 | 0 | 10 |

## 7.2. Proof of Concept: Validation and Further Work

Predictive ability of the welded connection identification method using YOLOv5s far surpassed that of the Hough Circle Transform method. In Table 16, YOLOv5s is validated against the proof-of-concept requirements.

*Table 16 YOLOv5s method validated against the proof-of-concept requirements*

| Index | Requirement | Pass (Y/N) | Evaluation |
|---|---|---|---|
| 1 | Identify all milling locations within a 3mm radius of the ground truth | Y | The best performing model developed in this research was trained on modules 1-3, tested on module 4, and correctly identified all milling locations at the threshold of 3mm. Results summarised in Table 15. |
| 2 | Make location inferences at least 10x faster than a human labeller | Y | Compared to times recorded in Appendix J, YOLOv5s' inference was always at least 200x faster than human labellers. The code written to save the co-ordinate data took considerably longer but was still 11.4x faster on average. Code optimisation would improve on this time. |
| 3 | Quickly configurable for new module types – preferably take no time at all to reconfigure | Y | The best performing model proves that it is possible for the vision system to develop and abstract concept of a cell. The best model trained on modules 1-3 was able to locate all cells in the unseen module 4, taking no time to reconfigure. |
| 4 | Use standard hardware and equipment | Y | Images were captured on a smartphone and the model was trained at 640x640 to validate that the technique could work on standard machine vision camera equipment (5.4.5). Training was carried out on GPUs accessible through Colab, and all code was written on a personal laptop. |
| 5 | Integrate with robotic operating system and develop motion planning capability | N | It was not possible to integrate the output of welded connection identification data with a robotic operating system software or simulate motion planning in this research project. The output of the method is a set of co-ordinates based on imagery input to the model. Co-ordinates are the basis upon which robotic systems move in space, so it is technically feasible to implement. |

Further Work

Despite meeting 4/5 of the requirements specified in section 3.2.2, it was not possible to simulate a robotic motion planner using the output of the prediction algorithm in this research. The objective of further research to address this problem should focus on validating that with a full set of accurate predictions in x-y pixel space, it is possible to define a set of co-ordinates in mm space that a robotic arm can move through. The central technical challenge of this objective would be to handle the variability of proximity of the cells to the camera, which would vary with module depth, by automatically interpreting the appropriate spatial resolution (mm: pixel ratio) to generate a path for the effector in spatial coordinates.

This challenge is related to a central assumption made in the development of this vision system, specified in section 3.2.1: that it is possible to control the depth to which the milling piece cuts into the welded connections so as to minimise unnecessary damage to the cells. This would require sensitive monitoring and feedback, and the development of a control system to manage this is of high importance to the success of a fully automated system. The assumption that the location in which to cut is the centre of the cell, given tooling with an appropriate cutting radius, should also be validated. Module 1 presents an example of a cell connection design that could prove incompatible with this assumption.

Additionally, further research should be carried out to identify the point at which the model begins to overfit to module designs contained in the training set, so that it does not lose its capability to generalise to unfamiliar cell connections.

# 8. Conclusion

This project investigated ways to address the challenge of developing technology to support the automated disassembly of LIB modules from EVs. An initial research phase set out to develop an understanding of the complexity and variety of LIB module designs, the tools and processes required to disassemble them, and how automated disassembly techniques from other fields could be applied in context.

Key findings from this phase included analyses of the breadth of module design [6], and existing studies in manual and automated disassembly of electronic and EV components, including LIBs [10] [11]. A framework of external and system factors influencing the capability of automated disassembly [14] led to the conclusion that improved techniques in robotic cognition and perception are key requirement of automated disassembly systems, especially those required to act in the face of complexity, variety, and uncertainty confronted when disassembling a diverse set of products, as is the case with LIB modules.

Computer vision techniques have been implemented in the context of automated disassembly to identify key components such as LCD screens [20] and screws [21], as well as assisting in planning disassembly routines for a single LIB design [12]. Other than unscrewing connectors, no research could be identified in the development of a computer vision system that can assist in a task of LIB module disassembly across a breadth of module designs.

Development of a novel motion planning method for the disassembly of welded connections across the product family of cylindrical cell modules was established as the objective for the remainder of research. 25 images each of 4 unique module designs were collected and labelled. Two computer vision methods were tested in their capability to detect welded connections and evaluated against ground truth labels: the Hough Circle Transform, and object detection with the YOLOv5s model architecture. With the right training configuration, YOLOv5s was able to identify all cell connections on a module when trained on that module's data.

Following further experimentation, a model was trained on images of 3 modules, and was able to produce perfect predictions for the fourth, unseen module, as defined by the requirements of the system. This result implies that with enough training on representative data, ensuring not to overfit, this method would be able to generalise to all cylindrical cell connection variants and develop motion plans for this aspect of module disassembly a priori. Without any manufacturer's data, manual instruction, or rigid disassembly systems, a vision system based on the methods developed in this proof-of-concept would be fully autonomous in the disassembly of cell connections.

By experimenting with the size of dataset required for model training, it was demonstrated that 10 labelled images are sufficient to teach the model an entirely new representation of cell connections, such that it can generate perfect predictions on different images of the same module. This suggests that it would be a cost and time effective process to retrain the model to recognise cell connection designs that it previously could not.

Further work objectives have been identified to address the unmet objectives of this research and could be the basis of a year 5 design project. The project's focus might be to implement a vision system and motion planner into a fully automated robotic station equipped to recognise and disassemble a variety of welded cell connections. All remaining objectives have been met through the development of a vision system capable of generalising its ability to locate cell connections on unseen designs with speed and precision. The results of this research verify that with deep learning, it is possible to endow robotic systems with the perceptive capability to deal with variations and uncertainties in the disassembly of a variety of LIB modules.

# References


[1] J. Peters and et al, “The environmental impact of Li-Ion batteries and the role of key parameters – A review,” *Renewable and Sustainable Energy Reviews,* pp. 419-50667, 2015.

[2] statista, “Projected size of the global electric vehicle fleet between 2021 and 2025,” 2 February 2022. [Online]. Available: https://www.statista.com/statistics/970958/worldwide-number-of-electric-vehicles/. [Accessed 21 March 2022].

[3] G. Harper, “Recycling lithium-ion batteries from electric vehicles,” *Nature,* vol. 575, p. 75–86, 2019.

[4] L. Ahmadi, S. Young, M. Fowler and et al, “A cascaded life cycle: reuse of electric vehicle lithium-ion battery packs in energy storage systems,” *Int J Life Cycle Assess,* vol. 22, p. 111–124, 2017.

[5] J. Pearson, “Engineering Design Project Brief: Automating Battery Recycling,” 2021.

[6] S. Arora, in *Behaviour of Lithium-Ion Batteries in Electric Vehicles*, Springer, Cham, 2018, pp. 175-200.

[7] E. Gerlitz, “Analysis of the Variety of Lithium-Ion Battery Modules and the Challenges for an Agile Automated Disassembly System,” in *8th CIRP Global Web Conference – Flexible Mass Customisation*, 2021.

[8] J. Diekmann, “Ecological recycling of lithium-ion batteries from electric vehicles with focus on mechanical processes,” *J. Electrochem,* pp. A6184-A6191, 2017.

[9] T. Elwert, Behaviour of Lithium-Ion Batteries in Electric Vehicles, Springer, Cham, 2018.

[10] K. Wegener, “Disassembly of Electric Vehicle Batteries Using the Example of the Audi Q5 Hybrid System,” in *Conference on Assembly Technologies and Systems*, 2014.

[11] K. Wegener, “Robot Assisted Disassembly for the Recycling of Electric Vehicle Batteries,” in *The 22nd CIRP conference on Life Cycle Engineering*, 2015.

[12] M. Choux, “Task Planner for Robotic Disassembly of Electric Vehicle,” *Metals,* vol. 11, no. 3, 2021.

[13] C. Rujanavech and et al, “Liam - An Innovation Story,” September 2016. [Online]. Available: https://www.apple.com/environment/pdf/Liam_white_paper_Sept2016.pdf. [Accessed 23 March 2022].

[14] G. Foo, “Challenges of robotic disassembly in practice,” in *CIRP Life Cycle Engineering Conference*, 2022.

[15] H. Poschmann, “Disassembly 4.0: A Review on Using Robotics in Disassembly Tasks as a Way of Automation,” *Chemie, Ingenieur, Technik,* vol. 92, no. 4, pp. 341-359, 2020.

[16] R. Knoth, “Automated disassembly of electr(on)ic equipment,” in *Electronics and the Environment*, 2002.

[17] S. Vongbunyong, “Application of cognitive robotics in disassembly of products,” *CIRP Journal of Manufacturing Science and Technology,* vol. 62, pp. 31-34, 2013.

[18] C. Herrmann and et al, “Scenario-based Development of Disassembly Systems for Automotive Lithium Ion Battery Systems,” *Advanced Materials Research,* vol. 907, pp. 391-401, 2014.

[19] P. Viola and M. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features,” *COMPUTER VISION AND PATTERN RECOGNITION,* 2001.

[20] S. Vongbunyong, “Basic behaviour control of the vision-based cognitive robotic disassembly automation,” *Assembly Automation,* vol. 33, no. 1, pp. 38-56, 2013.

[21] D. Brogan, “Deep learning computer vision for robotic disassembly and servicing applications,” *Array,* vol. 12, no. 1, 2021.

[22] R. Duda and P. Hart, “Use of the Hough transformation to detect lines and curves in pictures,” *Communications of the ACM,* vol. 1, no. 1, pp. 11-15, 1972.

[23] “18650 remove spot weld tab,” 1 December 2019. [Online]. Available: https://www.youtube.com/watch?v=6av-BW0DoM4&t=3s&ab_channel=IamyourDoddfather. [Accessed 3 March 2022].

[24] S. Russel and P. Norvig, Artificial Intelligence: A Modern Approach, Hoboken, New Jersey: Pearson, 2021.

[25] microsoft, “Collect images,” microsoft, 03 March 2022. [Online]. Available: https://docs.microsoft.com/en-us/ai-builder/collect-images. [Accessed 17 April 2022].

[26] S. Bahrampour, “Comparative Study of Deep Learning Software Frameworks,” *arXiv,* 2015.

[27] A. Karpathy, “A Recipe for Training Neural Networks,” 25 April 2019. [Online]. Available: http://karpathy.github.io/2019/04/25/recipe/. [Accessed 5 April 2022].

[28] Wikipedia, “Hyperparameter optimization,” 9 April 2022. [Online]. Available: https://en.wikipedia.org/wiki/Hyperparameter_optimization. [Accessed 24 April 2022].

[29] G. Bradski, Learning OpenCV, Sebastopol, CA: O'Reilly, 2008.

[30] F. Newport-Mangell, “design-project-4,” 2022. [Online]. Available: https://github.com/fxn-m/design-project-4.

[31] rapidtables, “pixel-ruler,” [Online]. Available: https://www.rapidtables.com/web/tools/pixel-ruler.html. [Accessed 18 April 2022].

[32] Z. -Q. Zhao, “Object Detection With Deep Learning: A Review,” *IEEE Transactions on Neural Networks and Learning Systems,* vol. 30, no. 11, pp. 3212-3232, 2019.

[33] L. Tsung-Yi and et al, “Microsoft COCO: Common Objects in Context,” *arXiv,* 2015.

[34] electricbike.com, “Introduction to battery pack design and building, Part-2,” electricbike.com, 30 March 2019. [Online]. Available: https://www.electricbike.com/introduction-battery-design-2/. [Accessed 3 April 2-22].

[35] S. Fortmann-Roe, “Understanding the Bias-Variance Tradeoff,” June 2012. [Online]. Available: http://scott.fortmann-roe.com/docs/BiasVariance.html. [Accessed 23 April 2022].